PYTHON PROGRAMMING

UNIT - 1

Introduction and basics

Python is a type of programming language.
It is a high-level interpreted and object oriented programming (OOPs) language.
It is easy to use and is readable and it has a variety of application such as software development, game development etc.

Disadvantage :- It is not fastest language.
It is a case sensitive language.

The code that we write in the program is converted into computer language (0101).

The programm that convert the code in computer language is called interpreter. Thats why Python is an interpreted language.

The extension of python file is .py

To

IDEs

It is referred as integrated development environment
It is an interface that provide the user to write his/her codes ~~which~~ / in to interact with the PC.

It provides the code to be attractive.

There are many types of IDEs such as

Pycharm, Spyder, PyDev, Atom, Visual Studio.

We can check errors easily by this.

PYTHON DECORATOR

It allows programmer to modify the behaviour of a function / or class.

VARIABLES

Variables are like containers that hold some data / values

Process of writing a variable is called declaring a variable.

Ex: x = 20

x holds 20 in it.

DATA TYPES

The data stored is in different types.

(i) Numeric :- Integer , float
20 20.5

(ii) String :- Word or sentence in double or single quote
"Hello world"

(iii) List :- A list contain no. of items.
It is ordered and indexible.

Syntax- x = ["Hello", 1, 4]

To concatenate we (+), to repeat we use (*) and to slice (:)

(iv) Tuple :- It is just like list.
Difference is that the items can not be changed.

Syntax - x = ("Hello", 1, 4)

(v) Dictionary :- Collection of key value - pairs.
These can be of any type.

Syntax - y = {"Name": "Shruti", "Date": 18}

To access the item
y ["Name"]
Output = 'Shruti'

(vi) Boolean :- Data stored in the form of True or False

Ex- a = True
b = False

TYPE CASTING / TYPE CONVERSION

Process of converting one type of data into another.

a) Implicit type conversion :- Data type conversion takes place during compilation.

It is handled by Python itself.

Ex- a = 5
b = 5.5
sum = a + b
print (sum)

Output - 10.5 float, here a is taken as float by Python itself.

b) Explicit type Conversion :- It is also known as type casting.

In this the programmer defines the type in which the data is to be converted.

Ex - x = 20.5

```
y = int(20.5)
print(y)
```

Output :- 20 which is integer type

Comments

This is a block of statement that is used to describe or explain the programme or code.

A comment starts with "#"

Ex- Sum = a + b # Addition of two no.

The python ignores whatever written after the # and show no error.

It is also used to prevent execution of some code.

There is no multiline comment so for multiline comment we use triple quotes (''' ''') as python ignores string literals that are not assigned to a variable.

Identifiers and Keywords

Identifiers are the name given to a variable, function, class etc.

Rules to name a variable :-

A variable name must start with a letter or the underscore character (_).

It cannot start with with a number.

A variable can only contain alpha-numeric character and underscore (A-z, 0-9, _)

These are case sensitive
Age, AGE, age are three different variables

Three type of Cases :-

Camel Case :- Each word, except the first starts with capital letter

myVariableName = "Vidyarthi"

Pascal Case :- Each word starts with capital letter.

MyVariableName = "Vidyarthi"

Snake Case :- Each word separated by underscore (_).

my_variable_name = "Vidyarthi"

Note :- Space is not accepted in writting variable names.

→ Assigning value to multiple variables

x, y, z = "Vidyarthi", "Python", "Shruti"

→ Assigning one value to multiple variable

x = y = z = "Vidyarthi"

Keywords :- It is a reserved word which we can't use as a name of our variable, class, function etc.

Example :- and, as, assert, class, try, def etc.

EXPRESSIONS

It is combination of variables, operators that evaluates to a value.

The interpreter interprets this value and assign this value to variable given.

Ex - expression = 4 + (3 * 2)

Output

print(expression)

Output :- 10

Operators :

It is the symbol used to perform operation on variables or values.

Operator taking one operand called unary operator.
Operator taking two operand called binary operator.

Types of Operators :

(i) Arithmetic operators :- Used to perform arithmatic operations like addition, subtraction etc.

+ : Adds operands
- : Subtracts operands
* : Multiplies "
/ : Divides "
** : Exponential operands increases in terms of power
% : Modulus operand to find remainder
// : Division operator to find quotient and remove fractional part.

(ii) Assignment Operator :- Used to store right side operands to left side operand or to assign value to variable.

= : Store right side value to left side variable or right side operand to left side operand

a = b+c

+= : Adds right operand to left and store value in left operand.
a+=b or (a=a+b)

-= : Same as addition but it subtracts.
a-=b (a=a-b)

*= : Work same as above but it multiplies.
a*=b (a=a*b)

/= : Work same as above but it divides
a/=b (a=a/b)

%= : Work same as above but it gives the remainder
a%=b (a=a%b)

**= : Work same but it gives the raised exponential value.
a**=b (a=a**b)

//= : Work same but gives the quotient without fraction part.
a//=b (a=a//b)

(iii) Bitwise Operator :- It performs bitwise operation on 0101 kind of operands.

& Bitwise AND — AND operation on operands x & y

| Bitwise OR — OR operation on operands x|y

^ Bitwise XOR — XOR operation on operands
x ^ y

~ Bitwise inverse — Used to inverse operands or give NOT of the operand
~x

<< left shift — Shift bits to left
x << 2 (Ex if x = 0101 result 010100)

>> right shift — Shifts bits to right
x >> 2 ~~(0001)~~ (0010) if x = 1010

(iv) Comparison Operator :- Used to compare ~~two~~ values or operands.

It result is always true or false

== : Checks for equality 10 == 20, result False
!= or <> : Checks for inequality 10 != 20, result True
> : For greater than 10 > 20, result False
< : For less than 10 < 20, True
>= : For greater than or equal to 10 >= 20 False
<= : For less than and or equal to 10 <= 20 True

(v) Identity Operator :- Used to check if the ~~two operands~~ operands are same or not.

is : returns true if operands are same and false if not same.
Ex :- if 10 is 20 results false

is not is : return false if operand are same and true if not same.

10 is not 20, true

(vi) Logical Operators :- Used to check conditions. The result of this is either true or false. Suppose x = true, y = False

and : Performs AND operation, if both operands are true, it returns true.
x and y results False

or : Perform OR operation, if any operand is true, returns true.
x or y results True

not : Performs NOT operation
not x results False

(vii) Membership Operator :- Checks if the value or operand is a part of other value or operand.
It checks whether an element is present in the given item or not.

Let x = [20, 30, 40] is a list and y = 30

in : Returns true if y is in list x and false if not
y in x, True

not in : returns true if y is not in x and false if it is in x

y not in x , False

Operator Precedence

In an expression with more than one operator we need to identify the correct sequence to evaluate the expression because result changes depending upon the precedence or importance.

Ex :- 10 + 5 / 5

if we go from left to right result is 3 and if we go from right to left result is 11.

So the value changes thats why we need precendence The precendence is given from high to low :-

** > (*, /, %, //) > (+, -) > (<<, >>) > & > (^, |) > (<=, <, >, >=) > (<>, ==, !=) > (=, %=, /=, //=, -=, +=, *=, **=) > (is, is not) not is) > (in, not in) > (NOT, OR, AND)

Operator Associativity

It decides the order in which the operators with same precedence are executed.

In python, it has left to right associativity. So if, we have multiple operator with same precedence then the execution is done from left to right.

Ex :- 3 * 4 // 6

Output :- 2

Programm to Print even length words in a string

```
input_string = input("Enter the string")
list_of_words = input_string.split()
for word in list_of_words:
    if len(word)%2 == 0:
        print(word)
```

UNIT - 2

Conditional Statement in Python

These are the statements used to make decision by using certain conditions.

The condition is specified in the programm which give results as true or false.

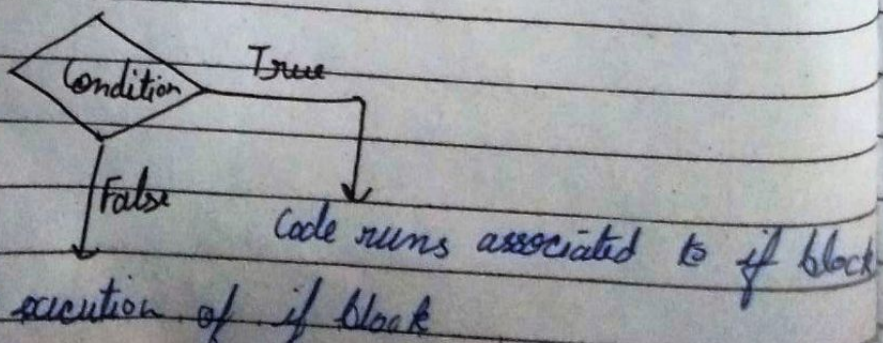
On the satisfaction of these condition the block of code associated with it runs

Types of Conditional Statements :-

(i) If statement :- It consist of a boolean expression that results true or false followed by if this condition becomes true, the block of code associated with it runs otherwise not.

Syntax :- if (Boolean expression):
    Block of code

Flow Chart :-



Ex :- variable = 100
if (var == 100):
    print ("value is 100")
print ("bye")

Output :- Value is 100
Bye

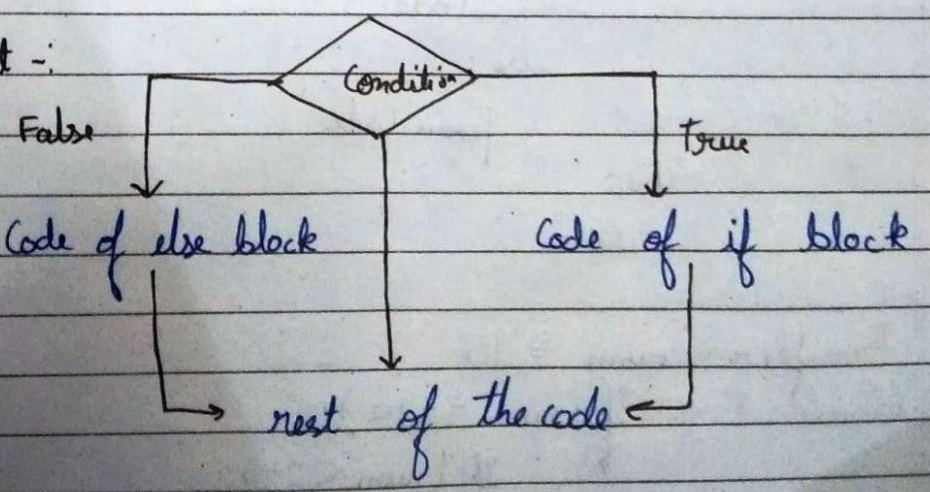
(ii) If - else statement :- It is like if statement, only difference is that it has a else block also along with if block.

So, when if condition is not satisfied then it runs the code in else block.

Syntax :-

```
if (Boolean expression):
    Code
else:
    Code
```

Flow Chart :-



Ex- num = 5

```
if (num > 10):
    print ("number is greater than 10")
else:
    print("number is less than 10")

print ("bye")
```

Output: number is less than 10
bye

(iii) Nested if statement: These statement are nested inside other if statement. In other words an if statement inside an if statement.

Used to check secondary condition when one condition is satisfied.

Syntax:

```
if (Boolean expression 1):
    if (Boolean expression 2):
        Code
    else:
        print Code
else:
    Code
```

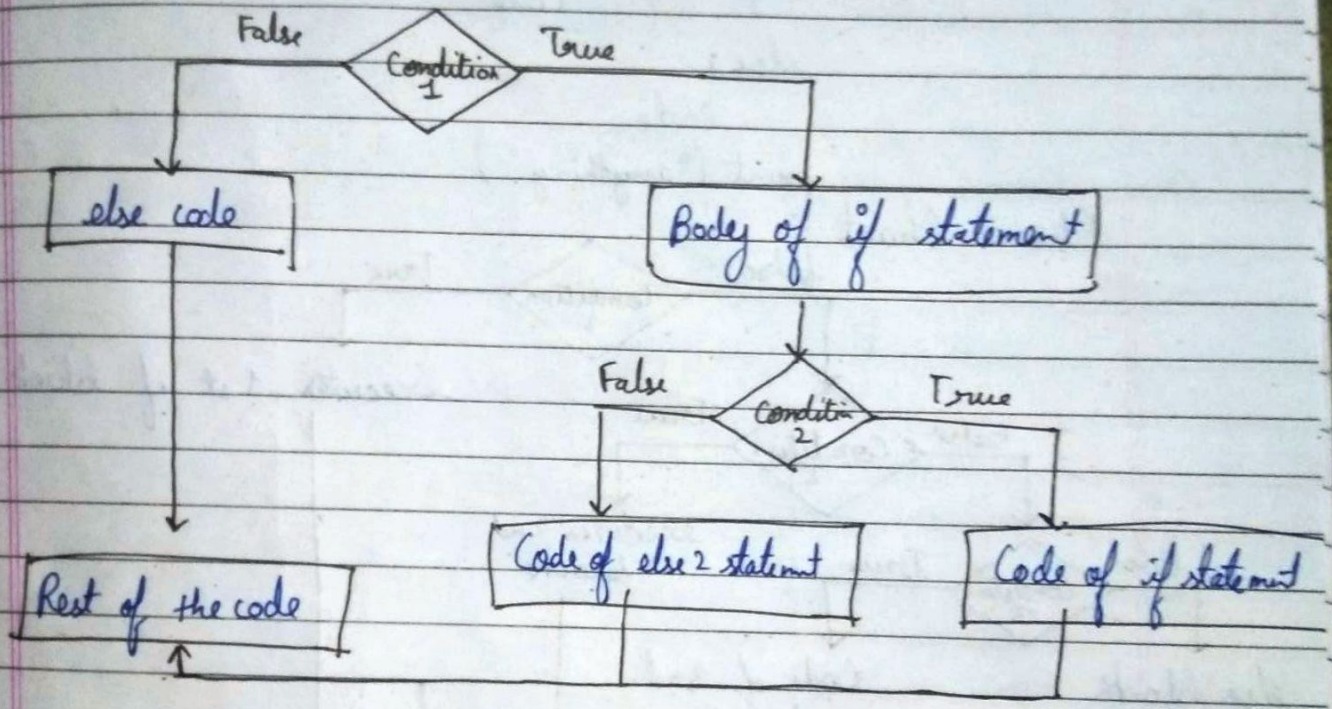
Example:

```
num = 10
if (num == 10):
    if (num < 20):
        print ("number less than 20")
    else:
        print ("number not less than 20")
else:
```

print ("number not equal to 10")

print ("bye")

Flow Chart :-



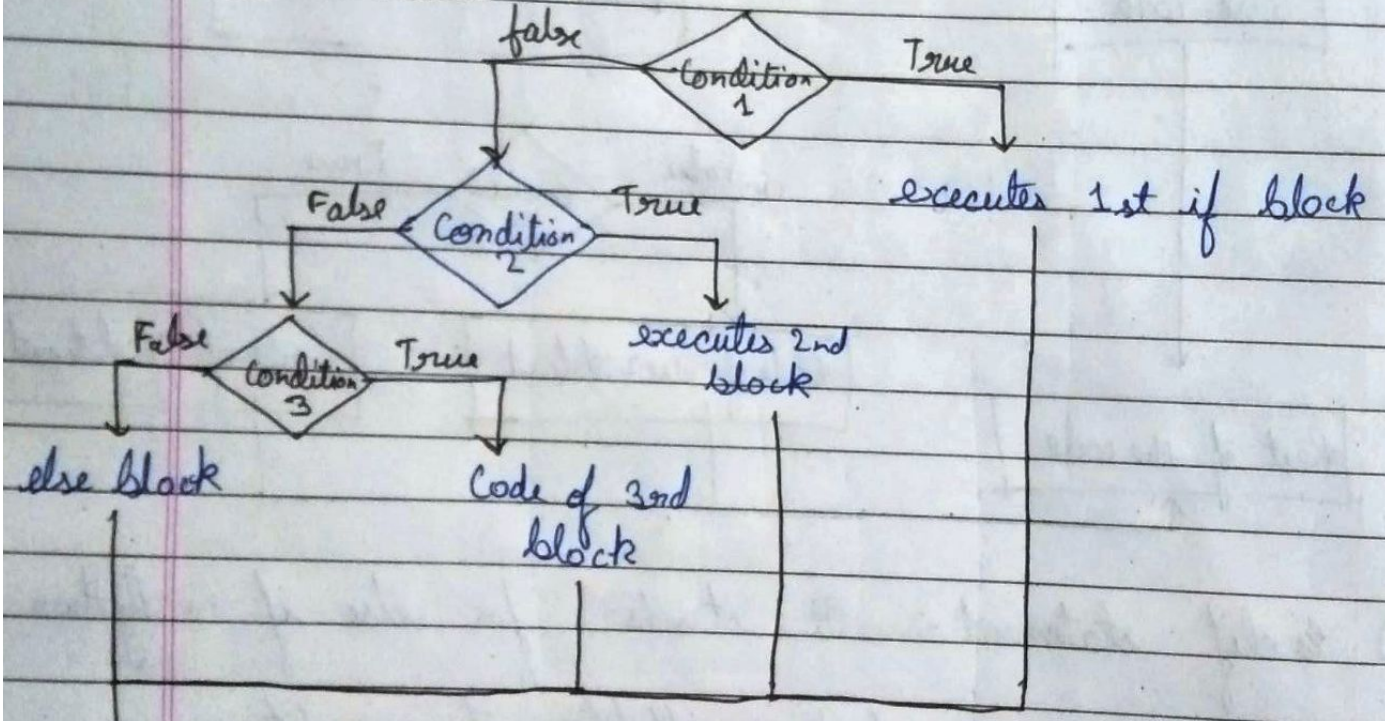
(iv) elif statement :- It stands for else - if in Python. It is used to check multiple if conditions.

It checks first condition if it satisfies then the code of first condition is executed and if not then it checks other condition and process goes on untill one condition satisfies and if no condition satisfies it executes the else block.

Syntax :-

```
if (expression 1):
    Code
elif (expression 2):
    Code
elif (expression 3):
    Code
else:
    code
print ("anything")
```

Flow Chart :-



Questions :-

Write program :- ① Check no. is even or odd

" " " " greater among three no.s

Expression Evaluation

Actions are performed in two forms

→ Expression evaluation, where it returns a value. and,

→ Statement execution, where it does not return any value.

Float Representation

Float represent floating-point number in python.

They represent the real number or number in desimals.

Ex- 20.8, 3.6

Maximum value any floating point number can be is approx 1.8×10^{308}

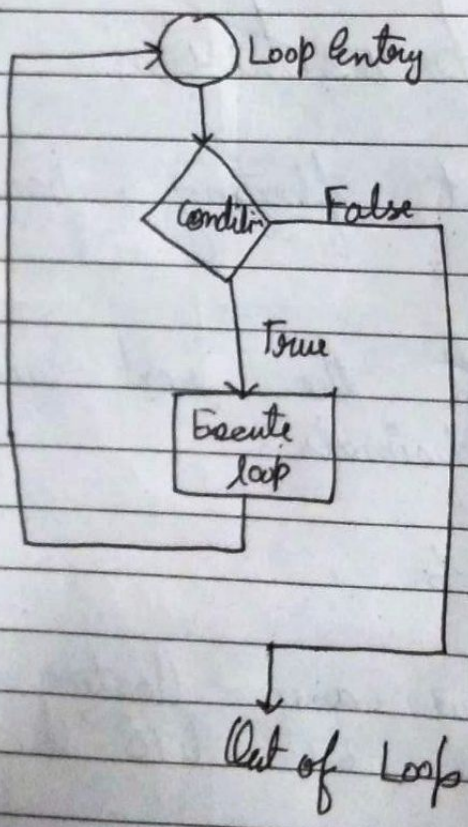
No. greater than this is indicated by a string inf in python.

LOOPS

It is a programming structure that repeats the code inside it untill the described condition is met.

It is used to execute block of code multiple times or the no. of times we want

Flow Chart



Syntax

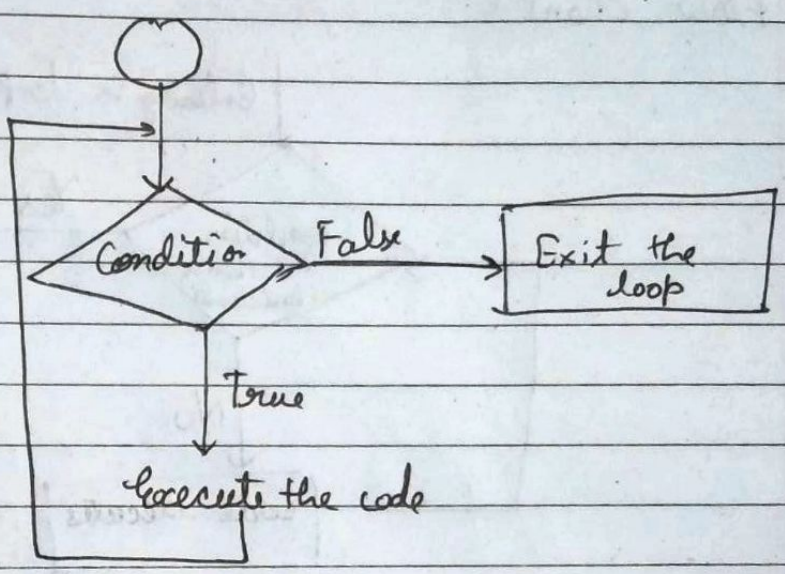
TYPES of LOOPS

(1) While Loop :- It is used to repeatedly execute a programm untill the condition provided is true.

It lets to iterate over a block of code.

Syntax :- ~~while~~ while (condition):
code

Flow chart



Ex :- count = 0
while (count < 9):
print ("The count is", count)
count = count + 1

Output :- The count is 0
" " " 1
" " " 2
; ; ;
; ; ;

(ii) For loop :- It is used to execute a block of code untill the given condition becomes false.

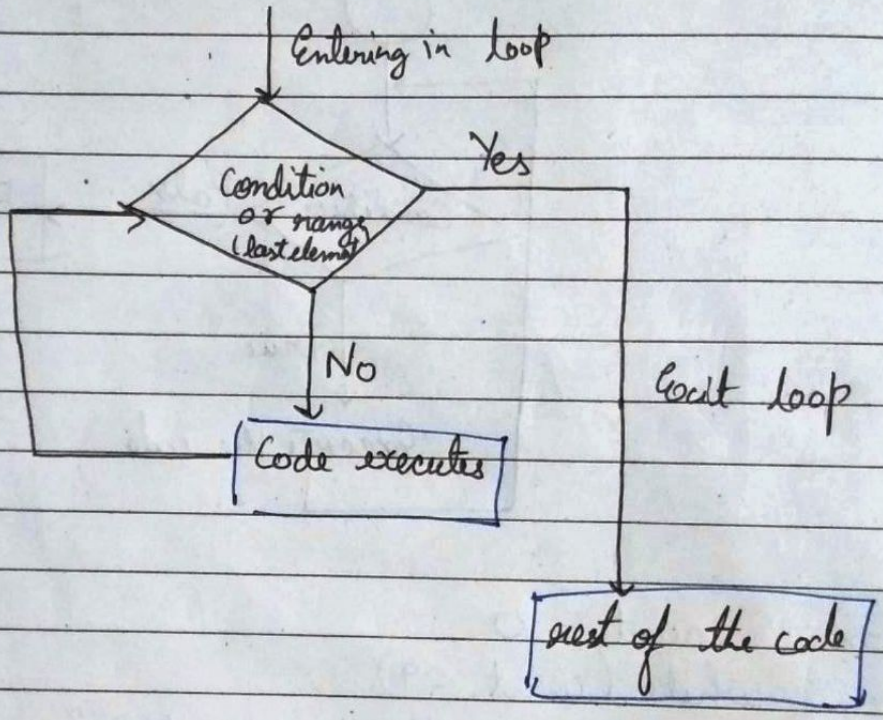
It is used when we know the number of

time to iterate or for a range of numbers.

Syntax:

```
for item in sequence or range:
    code
```

Flow chart:-



For example:-

```
i = 1
for i in range (1, 8):
    print 2*i
```

Output:- 2
4
6
8
10
12
14

(iii) Nested loop :- Loop inside other loop is called nested loop.

Syntax :-
```
for (condition 1):
    for (condition 2):
        Code
```

Ex :-
```
for i in rang (1, 9)
    for j in range (i)
        print (i, end = ' ')
    print ()
```

Output :-
```
1
2 2
3 3 3
4 4 4 4
```

Break Statement

The break keyword terminates the loop and transfer the control to the end of the loop.

It exits the loop and execute the code just after the loop.

Ex :-
```
count = 2
while True:
    print (count)
    count = count + 2
```

```
if count >= 12:
    break
```

Output :
2
4
6
8
10

Continue Statement

It causes execution to immediately continue at the start of the loop, it skips the execution of the remaining part of loop.

It terminates the ongoing iteration and move the control to the start or top of the loop and the condition is again evaluated.

Ex :
```
for i in range(1, 10):
    if(i%2 != 0):
        continue
    print(i)
```

Output :
2
4
6
8